



**LEGAL NOTICE:**

**© Copyright 2007 - 2015 NVM Express, Inc. ALL RIGHTS RESERVED.**

This NVM Express revision 1.2 erratum is proprietary to the NVM Express, Inc. (also referred to as "Company") and/or its successors and assigns.

**NOTICE TO USERS WHO ARE NVM EXPRESS, INC. MEMBERS:** Members of NVM Express, Inc. have the right to use and implement this NVM Express revision 1.2 erratum subject, however, to the Member's continued compliance with the Company's Intellectual Property Policy and Bylaws and the Member's Participation Agreement.

**NOTICE TO NON-MEMBERS OF NVM EXPRESS, INC.:** If you are not a Member of NVM Express, Inc. and you have obtained a copy of this document, you only have a right to review this document or make reference to or cite this document. Any such references or citations to this document must acknowledge NVM Express, Inc. copyright ownership of this document. The proper copyright citation or reference is as follows: "**© 2007 - 2015 NVM Express, Inc. ALL RIGHTS RESERVED.**" When making any such citations or references to this document you are not permitted to revise, alter, modify, make any derivatives of, or otherwise amend the referenced portion of this document in any way without the prior express written permission of NVM Express, Inc. Nothing contained in this document shall be deemed as granting you any kind of license to implement or use this document or the specification described therein, or any of its contents, either expressly or impliedly, or to any intellectual property owned or controlled by NVM Express, Inc., including, without limitation, any trademarks of NVM Express, Inc.

**LEGAL DISCLAIMER:**

THIS DOCUMENT AND THE INFORMATION CONTAINED HEREIN IS PROVIDED ON AN "**AS IS**" BASIS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, NVM EXPRESS, INC. (ALONG WITH THE CONTRIBUTORS TO THIS DOCUMENT) HEREBY DISCLAIM ALL REPRESENTATIONS, WARRANTIES AND/OR COVENANTS, EITHER EXPRESS OR IMPLIED, STATUTORY OR AT COMMON LAW, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, VALIDITY, AND/OR NONINFRINGEMENT.

All product names, trademarks, registered trademarks, and/or servicemarks may be claimed as the property of their respective owners.

NVM Express Workgroup  
c/o Virtual, Inc.  
401 Edgewater Place, Suite 600  
Wakefield, MA 01880  
info@nvmexpress.org

## NVM Express™ Technical Errata

<b>Errata ID</b>	<b>005</b>
<b>Revision Date</b>	<b>6/3/2015</b>
<b>Affected Spec Ver.</b>	<b>NVM Express™ 1.2</b>
<b>Corrected Spec Ver.</b>	

### Errata Author(s)

Name	Company
Amber Huffman, Matthew Wilcox, Jonathan Hughes, John Carroll	Intel
Judy Brock	Samsung
Peter Onufryk	PMC
Jim Hatfield	Seagate
Naren Nadesan	HGST
Lee Prewitt	Microsoft

### Errata Overview

This document clarifies namespace concepts. The concepts of active and inactive namespace IDs at a controller level are clarified. The ECN introduces the concept of allocated and unallocated namespace IDs at an NVM subsystem level to indicate whether a namespace exists.

## Revision History

Revision Date	Change Description
2/3/2015	Draft material to clarify the namespace concepts.
2/25/2015	Added concept of full/empty NSID at NVM subsystem level.
3/4/2015	Used allocated/unallocated NSID to specify whether a namespace exists. Other changes based on 2/26 discussion.
3/7/2015	Updated the definition of namespace identifier and the Identify data structure that specifies the CNS based operations.
4/8/2015	Updates based on discussion at 3/12 meeting. Added picture to clarify how NSID Types relate.
4/13/2015	Accepted red-lines and made further updates based on reflector discussion.
4/14/2015	Incorporated proposed changes from Naren, Jonathan, John, and Lee.
4/17/2015	Added that NSID FFFFFFFFh is a broadcast value that is used to specify all namespaces. Clarified recommendation that NSIDs should not be changed.
6/3/2015	Ratified.

## Description of Specification Changes

**Modify section 6.1 as shown below:**

### 6.1 Namespaces

A namespace is a collection of logical blocks that range from 0 to the capacity of the namespace – 1. ~~The number of possible namespaces present is reported in the Identify Controller data structure.~~ A namespace ID (NSID) is an identifier used by a controller to provide access to a namespace.

Valid NSIDs are the range of possible NSIDs that correspond to a namespace that may exist in the NVM subsystem. Any NSID is valid, except if it is zero or greater than the Number of Namespaces field reported in the Identify Controller data structure. NSID FFFFFFFFh is a broadcast value that is used to specify all namespaces. An invalid NSID is any value that is not a valid NSID or the broadcast value.

Active NSIDs are valid NSIDs that are attached to the specific controller. Valid NSIDs that are not attached to the specific controller are called inactive. An active NSID becomes inactive when the associated namespace is detached from the specific controller or is deleted.

Allocated NSIDs are valid NSIDs that refer to namespaces that currently exist within an NVM subsystem. An allocated NSID may not be attached to any controller. An allocated NSID shall be attached to a controller before host software may submit I/O commands for that namespace on that controller. An allocated NSID becomes unallocated when the associated namespace is deleted.

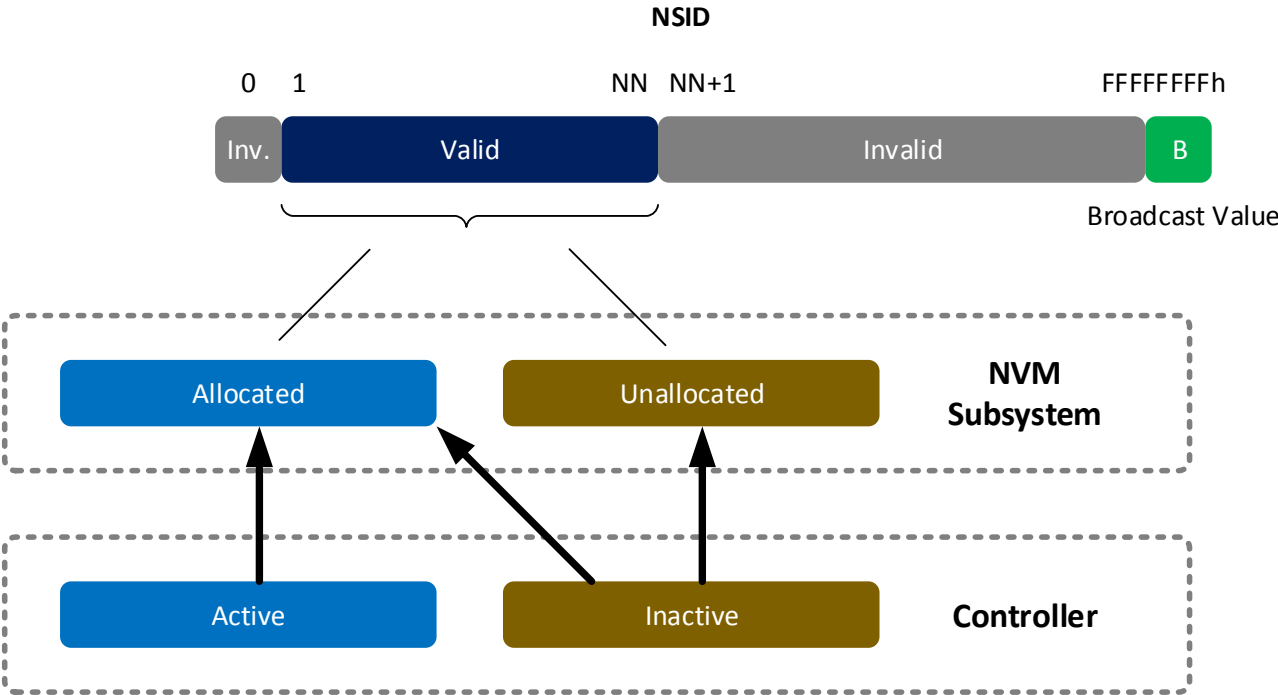
Unless otherwise noted, specifying an inactive namespace ID in a command that uses the namespace ID shall cause the controller to abort the command with status Invalid Field in Command. Specifying an invalid NSID in a command that uses the NSID field shall cause the controller to abort the command with status Invalid Namespace or Format.

The following table summarizes the valid NSID types. Figure **TBD** visually shows the NSID types and how they relate.

Valid NSID Type	The associated namespace
Active	is attached to this controller
Inactive	is not attached to this controller

Allocated	exists in the NVM subsystem
Unallocated	does not exist in the NVM subsystem

Figure TBD: NSID Types



If Namespace Management is supported then Namespace IDs shall be unique within the NVM subsystem (e.g., namespace ID of 3 shall refer to the same physical namespace regardless of the accessing controller). If Namespace Management is not supported then Namespace IDs for private namespaces are not required to be unique.

The Identify command may be used to determine the active NSIDs for a controller and the allocated NSIDs in the NVM subsystem.

To determine the active NSIDs for a particular controller, the host may follow either of the following methods:

1. Issue Identify with the CNS field set to 00h for each valid NSID (based on the Number of Namespaces value in Identify Controller). If a non-zero data structure is returned for a particular NSID, then that is an active NSID.
2. Issue Identify with a CNS field set to 02h to retrieve a list of up to 1024 active NSIDs. If there are more than 1024 active NSIDs, continue to issue Identify with a CNS field set to 02h until all active NSIDs are retrieved.

To determine the allocated NSIDs in the NVM subsystem, the host may Issue Identify with the CNS field set to 10h to retrieve a list of up to 1024 allocated NSIDs. If there are more than 1024 allocated NSIDs, continue to issue Identify with a CNS field set to 10h until all allocated NSIDs are retrieved.

<INSERT BLANK LINE>

~~Unless otherwise noted, specifying an inactive namespace ID in a command that uses the namespace ID shall cause the controller to abort the command with status Invalid Field in Command. Specifying an invalid namespace ID in a command that uses the namespace ID shall cause the controller to abort the command~~

~~with status Invalid Namespace or Format.~~ Namespace IDs may change across power off conditions or due to namespace management. However, it is recommended that namespace identifiers remain static in order to avoid issues with EFI or ~~OSes. OS drivers fast discovery processes.~~

The Namespace Size field in the Identify Namespace data structure defines the total size of the namespace in logical blocks (LBA 0 through n-1). The Namespace Utilization field in the Identify Namespace data structure defines the number of logical blocks currently allocated in the namespace. The Namespace Capacity field in the Identify data structure defines the maximum number of logical blocks that may be allocated at one time as part of the namespace in a thin provisioning usage model. The following relationship holds: Namespace Size  $\geq$  Namespace Capacity  $\geq$  Namespace Utilization.

A namespace may or may not have a relationship to a Submission Queue; this relationship is determined by the host software implementation. The controller shall support access to any valid namespace from any I/O Submission Queue.

**Modify section 1.6.18 as shown below:**

### **1.6.18 Namespace ID (NSID)**

An identifier used by a controller to provide access to a namespace. Refer to section 6.1 for the definitions of valid NSID, invalid NSID, active NSID, inactive NSID, allocated NSID, and unallocated NSID.

~~The NSID value of FFFFFFFFh is used to indicate all namespaces. An invalid namespace ID is a namespace ID whose value is zero or whose value is greater than the value reported by the Number of Namespaces (NN) field in the Identify Controller data structure. All other namespace IDs are valid. A valid namespace ID that maps to an attached namespace is an active namespace ID. A namespace shall be attached to a controller before host software may submit I/O commands for that namespace. A valid namespace ID that does not map to an attached namespace is an inactive namespace ID. A valid namespace ID becomes inactive when the associated namespace is detached or deleted.~~

~~If Namespace Management is supported then Namespace IDs shall be unique within the NVM subsystem (e.g., namespace ID of 3 shall refer to the same physical namespace regardless of the accessing controller). If Namespace Management is not supported then Namespace IDs for private namespaces are not required to be unique.~~

Modify Figure 86 as shown below:

Figure 86: Identify – Data Structure Returned

CNS Value	Definition
00h	<p>The Identify Namespace data structure is returned to the host for the namespace specified in the Namespace Identifier (CDW1.NSID) field <del>if it is an active NSID the namespace is attached to this controller</del>. If the specified namespace is <del>not an active</del> <b>an inactive NSID namespace ID</b>, then the controller returns a zero filled data structure.</p> <p>If the controller supports Namespace Management and CDW1.NSID is set to FFFFFFFFh, the controller returns an Identify Namespace data structure that specifies capabilities that are common across namespaces <b>for this controller</b>.</p>
01h	The Identify Controller data structure is returned to the host for this controller.
02h	<p>A list of 1024 namespace IDs is returned containing active <b>NSIDs namespace IDs attached to this controller</b> in increasing order that are greater than the value specified in the Namespace Identifier (CDW1.NSID) field of the command. The data structure returned is a Namespace List (refer to section 4.8).</p> <p>Controllers that support specification revision 1.1 or later shall support this capability.</p>
03h – 0Fh	Reserved
<b>Namespace Management</b>	
10h	A list of up to 1024 namespace IDs is returned to the host containing <b>allocated NSIDs namespaces that are present in the NVM subsystem</b> with a namespace identifier greater than the value specified in the Namespace Identifier (CDW1.NSID) field. <del>The namespaces may or may not be attached to controller(s)</del> .
11h	<p>The Identify Namespace data structure is returned to the host for the namespace specified in the Namespace Identifier (CDW1.NSID) field <del>if it is an allocated NSID. The namespace may or may not be attached to this controller</del>. If the specified namespace is <del>an unallocated NSID invalid</del> then the controller returns a zero filled data structure. If the specified namespace is <del>an invalid NSID invalid</del> then the controller shall fail the command with a status code of Invalid Namespace or Format.</p>
12h	A Controller List of up to 2047 controller identifiers is returned containing a controller identifier greater than or equal to the value specified in the Controller Identifier (CDW10.CNTID) field. The list contains controller identifiers that are attached to the namespace specified in the Namespace Identifier (CDW1.NSID) field.
13h	A Controller List of up to 2047 controller identifiers is returned containing a controller identifier greater than or equal to the value specified in the Controller Identifier (CDW10.CNTID) field. The list contains controller identifiers in the NVM subsystem that may or may not be attached to namespace(s).
14h – 1Fh	Reserved
<b>Future Definition</b>	
20h – FFh	Reserved

Modify Figure 92 (Identify Namespace) as shown below:

15:8	M	<p><b>Namespace Capacity (NCAP):</b> This field indicates the maximum number of logical blocks that may be allocated in the namespace at any point in time. The number of logical blocks is based on the formatted LBA size. This field is undefined prior to the namespace being formatted. This field is used in the case of thin provisioning and reports a value that is smaller than or equal to the Namespace Size. Spare LBAs are not reported as part of this field.</p> <p><del>A value of 0h for the Namespace Capacity indicates that the namespace ID is an inactive namespace ID.</del></p> <p>A logical block is allocated when it is written with a Write or Write Uncorrectable command. A logical block may be deallocated using the Dataset Management command.</p>
------	---	---

***Modify the first paragraph of section 5.12 (Namespace Attachment) as shown below:***

The Namespace Attachment command is used to attach and detach controllers from a namespace. The attach and detach operations are persistent across all reset events.

***Modify the first paragraph of section 5.13 (Namespace Management) as shown below:***

The Namespace Management command is used to manage namespaces, including create and delete operations. Note: The controller continues to execute commands submitted to I/O Submission Queues while this operation is in progress.